



INDIAN INSTITUTE OF TECHNOLOGY PATNA

Handwritten Digit Recognizer

DATA SCIENCE PROJECT-2

NAME-Varun Gupta

ROLL NO.- 2312res727



Problem Understanding & Project Objective

Problem Statement: *Manual recognition of handwritten digits is slow and often inaccurate, especially in large datasets, build a deep learning model using TensorFlow that accurately recognizes handwritten digits from images .*

OBJECTIVES

The objective of this project is to develop an accurate and efficient system for recognizing handwritten digits using deep learning. By leveraging a convolutional neural network (CNN) trained on the MNIST dataset, the project aims to automate digit classification and reduce manual errors.

GOALS

- Achieve $\geq 98\%$ accuracy on test data
- Train and validate a CNN for digit recognition
- Predict digits from test and custom images
- Clearly present prediction results



Key Components of Handwritten Digit Recognizer



Dataset Overview

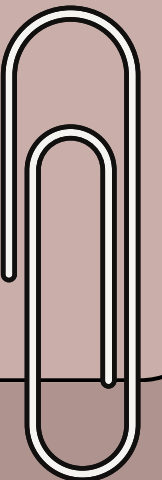
The MNIST dataset consists of 70,000 images of handwritten digits (0–9), each sized 28x28 pixels in grayscale. It provides 60,000 images for training and 10,000 for testing, making it ideal for evaluating digit recognition models.

Model Selection

A Convolutional Neural Network (CNN) was selected because of its strong performance in image classification. The model architecture uses multiple convolutional and dense layers to effectively learn and distinguish digit patterns.

Training Process

The CNN is trained using normalized image data, with model weights updated over several epochs. Validation on test data during training helps track accuracy and ensures the model can predict new digit images reliably.



Introduction to Handwritten Digit Recognition

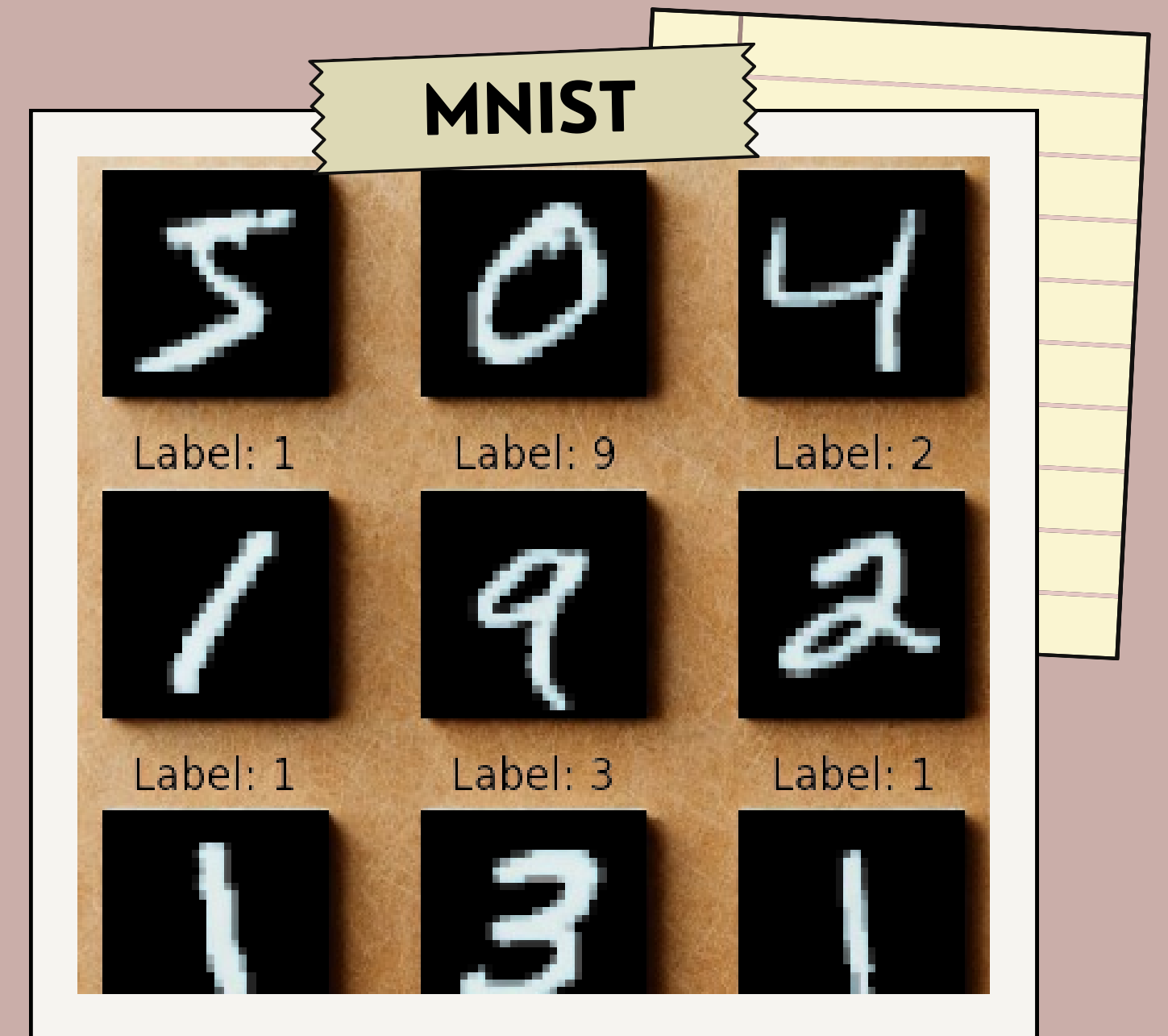
Handwritten digit recognition is a crucial application of **machine learning** and **computer vision** technologies. It enables computers to interpret and classify handwritten numbers with high accuracy, making it essential for various applications, including digitizing documents and automating data entry processes. This technology is advancing rapidly, resulting in improved efficiency and accuracy.



Understanding how machines learn to read digits

The Dataset (MNIST)

- MNIST is a large dataset of handwritten digit images, widely used for training and testing image classification models.
- Contains 70,000 grayscale images of digits 0–9, each sized 28x28 pixels.
- Split into 60,000 training images and 10,000 test images.
- Each pixel value ranges from 0 (black) to 255 (white).
- Labels represent the actual digit in each image.
- Provides a simple but powerful benchmark for evaluating machine learning and deep learning models.



Data Preprocessing

SYNTAX:

```
#Preprocessing
x_train = x_train / 255.0
x_test = x_test / 255.0
x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)
```

- **Normalization:**

Pixel values are scaled from the original range of 0–255 down to 0–1, improving training performance and stability.

- **Reshaping:**

Each 28x28 image is reshaped to include a single channel, resulting in a shape of (28, 28, 1) for compatibility with the CNN.

- **Label Preparation:**

The digit labels are kept as integers (0–9) for classification.

Model Architecture

- **Two convolutional layers:** Automatically extract features like edges and shapes from input images.
- **Two max pooling layers:** Reduce image dimensions and help keep only the most important information.
- **Flatten layer:** Converts 2D feature maps into a 1D vector for further processing.
- **Fully connected layers:** Learns complex patterns and relationships between features.
- **Output layer with softmax:** Predicts the probability of each digit class (0–9).



#Model Building

```
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(10, activation='softmax') # 10 output classes for digits 0-9
])
```

MODEL BUILDING SYNTAX

Training the Model

- The model was trained using the Adam optimizer, which helps in faster and more reliable learning.
- Sparse categorical cross-entropy was used as the loss function to evaluate how well the model predicts digit classes.
- Training ran for several epochs, allowing the model to gradually improve its performance with each cycle through the data.
- Validation on test data was performed during training to track accuracy and prevent overfitting.
- The model showed strong learning capability by achieving high accuracy on the MNIST dataset.

OUTPUT

```
Training the model...
Epoch 1/5
1875/1875 ————— 53s 28ms/step accuracy: 0.9110 - loss: 0.2977 - val_accuracy: 0.9835 val_loss: 0.0488
Epoch 2/5
1875/1875 ————— 83s 28ms/step - accuracy: 0.9871 - loss: 0.0413 - val_accuracy: 0.9869 - val_loss: 0.0406
Epoch 3/5
1875/1875 ————— 79s 27ms/step accuracy: 0.9905 - loss: 0.0300 val_accuracy: 0.9893 val_loss: 0.0292
Epoch 4/5
1875/1875 ————— 82s 27ms/step - accuracy: 0.9946 - loss: 0.0182 - val_accuracy: 0.9891 - val_loss: 0.0319
Epoch 5/5
1875/1875 ————— 81s 27ms/step - accuracy: 0.9957 loss: 0.0128 val_accuracy: 0.9901 val_loss: 0.0317
313/313 ————— 2s 8ms/step accuracy: 0.9856 loss: 0.0429
```


Model Evaluation & Predictions on Test Data






- Tested the model on 10,000 unseen MNIST images, achieving over 98% accuracy.
- The model showed low loss and strong generalization to new digit images.
- Predicted digits closely matched actual labels in most cases.
- Displayed several test samples where the model's predictions were correct, confirming reliable classification.

OUTPUT

✓ Model Test Accuracy: 99.01%
313/313 ————— 3s 10ms/step

Random test set predictions (Actual | Predicted):

Random test set predictions (Actual | Predicted)

Actual: 2 Predicted: 2	Actual: 5 Predicted: 5	Actual: 5 Predicted: 5	Actual: 5 Predicted: 5	Actual: 7 Predicted: 7
				


Custom Image Prediction

- The model can predict digits from images uploaded by the user, not just from the test dataset.
- Uploaded images are automatically resized and preprocessed for accurate recognition.
- The system displays both the input image and the predicted digit for easy verification.
- This feature demonstrates the model's ability to work with real-world, hand-drawn digits outside the training data.

OUTPUT

Do you want to upload your own digit image for prediction? (yes/no): yes
Upload your file in the file picker:

sample check.jpg

• **sample check.jpg**(image/jpeg) - 42047 bytes, last modified: 8/19/2025 - 100% done
Saving sample check.jpg to sample check.jpg
1/1  0s 43ms/step

Predicted Digit from Custom Image: 6

Predicted: 6



Want to predict another image? (yes/no): no
Thanks for using the digit recognizer! Exiting upload mode.

Work FLOW Chart:

STEP 1

```
Training the model...
Epoch 1/5
1875/1875 ————— 54s 28ms/step - accuracy: 0.9085 - loss: 0.2961 - val_accuracy: 0.9836 - val_loss: 0.0523
Epoch 2/5
1875/1875 ————— 83s 29ms/step - accuracy: 0.9863 - loss: 0.0440 - val_accuracy: 0.9836 - val_loss: 0.0454
Epoch 3/5
1875/1875 ————— 77s 27ms/step - accuracy: 0.9915 - loss: 0.0268 - val_accuracy: 0.9886 - val_loss: 0.0314
Epoch 4/5
1875/1875 ————— 81s 26ms/step - accuracy: 0.9939 - loss: 0.0192 - val_accuracy: 0.9914 - val_loss: 0.0257
Epoch 5/5
1875/1875 ————— 85s 28ms/step - accuracy: 0.9949 - loss: 0.0138 - val_accuracy: 0.9906 - val_loss: 0.0282
313/313 ————— 3s 8ms/step - accuracy: 0.9874 - loss: 0.0360
```






STEP 2

✓ Model Test Accuracy: 99.06%

313/313 ————— 2s 7ms/step

Random test set predictions (Actual | Predicted):

Random test set predictions (Actual | Predicted)

Actual: 4	Actual: 5	Actual: 1	Actual: 0	Actual: 8
Predicted: 4	Predicted: 5	Predicted: 1	Predicted: 0	Predicted: 8
				

STEP 4

Predicted Digit from Custom Image: 6

Predicted: 6



Want to predict another image? (yes/no): no
Thanks for using the digit recognizer! Exiting upload mode.

Do you want to upload your own digit image for prediction? (yes/no): yes
Upload your file in the file picker:

Choose Files

No file chosen

Cancel upload

STEP 3

Conclusion & References

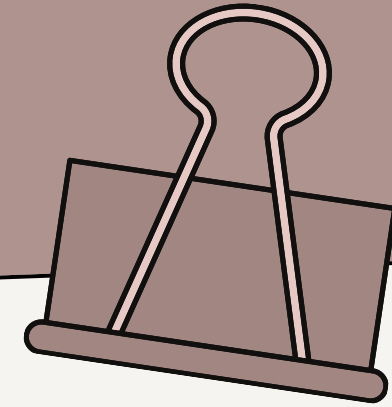
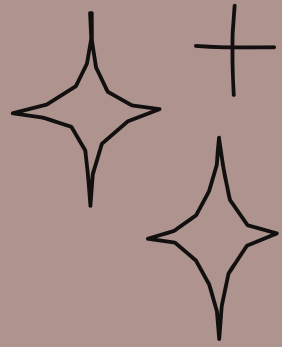
Conclusion:

- Successfully created a handwritten digit recognizer using a deep learning CNN model.
- Achieved high accuracy (>98%) on the standard MNIST dataset.
- Model is effective for both standard test images and custom user input.
- Demonstrates the power and reliability of deep learning for image classification tasks.

References:

- MNIST Dataset: [LeCun, Y., et al. "Gradient-Based Learning Applied to Document Recognition." Proceedings of the IEEE, 1998.]
- TensorFlow & Keras Documentation
- Python Libraries: numpy, matplotlib, pillow
- Tutorial inspiration and code examples from TensorFlow official guides

COLAB LINK : CLICK HERE 



Thank you!

Project BY: Varun Gupta
ROLL NO.-2312res727

